Introduction to Computer Science I

Switch Statement
Software Development Activities

Janyl Jumadinova

11 April, 2018
Control Structures

- Java programs are built from only these seven control structures:
  - *three selection* (if, if/else, switch)
  - *three repetition* (while, do/while, for)

- You implement computer algorithms by stringing sequences of these seven control structures together.
Selection

- if statement is a single-selection structure.
- if/else statement is a double-selection structure.
Selection

- if statement is a single-selection structure.
- if/else statement is a double-selection structure.
- What if you have a series of integral values you would like to test and you might possibly want to trigger multiple actions based on one value?
Selection

- if statement is a single-selection structure.
- if/else statement is a double-selection structure.
- What if you have a series of integral values you would like to test and you might possibly want to trigger multiple actions based on one value?
- A switch statement can re-implement most if or if/else structures more compactly.
- You can execute more than just one action with a switch, as opposed to the way a nested if/else structure works.
char character;
switch (character) {
    case 'a': // case labels
    case 'e': // separated by :
    case 'i': // character
    case 'o': // notice use of ‘ ’
    case 'u': // marks for char tests
        System.out.print (character+" is a lowercase vowel\n");
        break;
    default:
        System.out.print (character+" is not a lowercase vowel\n");
}
Switch Statement

- Control flow continues with the first statement following the switch block.
Switch Statement

- Control flow continues with the first statement following the switch block.
- The break statements are necessary because without them, statements in switch blocks fall through.
Switch Statement

- Control flow continues with the first statement following the switch block.
- The break statements are necessary because without them, statements in switch blocks fall through.
- All statements after the matching case label are executed in sequence, regardless of the expression of subsequent case labels, until a break statement is encountered.
Switch Summary

- if and if/else can test ranges of numbers using relational
  \((>, <, \geq \text{ and } \leq)\) and inequality \((! =)\) operators.

- switch statement can only make exact matches of values \((==)\).

- switch statement works with int, char, byte, short, String
  and some other special (enum) data types.

- if and if/else can test other data types such as floating point
  numbers.

- if and if/else can find one condition to be true and execute an
  action.

- switch statements find one match and continue executing code
  until a break is found.
Switch Summary

- **if and if/else** can test ranges of numbers using relational operators (>, <, ≥ and ≤) and inequality (!=) operators.
- **switch** statement can only make exact matches of values (==).
Switch Summary

- **if and if/else** can test ranges of numbers using relational (>), (<), (≥) and (≤) and inequality (≠) operators.
- **switch** statement can only make exact matches of values (==).
- **switch** statement works with `int`, `char`, `byte`, `short`, `String` and some other special (`enum`) data types.
Switch Summary

- **if and if/else** can test ranges of numbers using relational (>, <, ≥ and ≤) and inequality (! =) operators.
- **switch** statement can only make exact matches of values (==).
- **switch** statement works with int, char, byte, short, String and some other special (enum) data types.
- **if and if/else** can test other data types such as floating point numbers.
Switch Summary

- **if and if/else** can test ranges of numbers using relational
  ($>$, $<$, $\geq$ and $\leq$) and inequality ($!=$) operators.
- **switch** statement can only make exact matches of values ($==$).
- **switch** statement works with int, char, byte, short, String and some other special (enum) data types.
- **if and if/else** can test other data types such as floating point numbers.
- **if and if/else** can find one condition to be true and execute an action.
Switch Summary

- **if and if/else** can test ranges of numbers using relational (>), (<, ≥ and ≤) and inequality (! =) operators.
- **switch** statement can only make exact matches of values (==).
- **switch** statement works with int, char, byte, short, String and some other special (enum) data types.
- **if and if/else** can test other data types such as floating point numbers.
- **if and if/else** can find one condition to be true and execute an action.
- **switch** statements find one match and continue executing code until a break is found.
Object-oriented Design
Software Development

- Establishing the **software requirements**: *what* a program must accomplish.
Software Development

- Establishing the **software requirements**: *what* a program must accomplish.
- Creating a **software design**: *how* a program will accomplish its requirements.
Software Development

- Establishing the **software requirements**: what a program must accomplish.
- Creating a **software design**: how a program will accomplish its requirements.
- **Implementing** the design: process of writing the code (translating design into a programming language).
Software Development

- Establishing the **software requirements**: *what* a program must accomplish.
- Creating a **software design**: *how* a program will accomplish its requirements.
- **Implementing** the design: process of writing the code (translating design into a programming language).
- **Testing**: ensuring that a program will solve the intended problem.
Identifying Classes and Objects

- Identify the objects from the software requirements.
Identifying Classes and Objects

- Identify the objects from the software requirements.
- Decide what is an object and what is an attribute of another object.
Identifying Classes and Objects

- Identify the objects from the software requirements.
- Decide what is an object and what is an attribute of another object.
- Decide if need to have supporting class(es), in addition to classes that represent objects.
Identifying Classes and Objects

- Identify the objects from the software requirements.
- Decide what is an object and what is an attribute of another object.
- Decide if need to have supporting class(es), in addition to classes that represent objects.
- Assign responsibilities to each class (behaviors/methods of the class).